Redundancy Network Design for an Aircraft Door Management System

Redundancy Network Design for an Aircraft Door Management System

Lukas Schäfer[1,*], Sergio García[1],
Andreas Mitschke[2], Vassili Srithammavanh[3]

[1]School of Mathematics, University of Edinburgh
[2]AIRBUS Group Innovations, Germany
[3]AIRBUS Group Innovations, France

Abstract

The door management system (DMS) is a safety system in an aircraft which checks if all doors are properly closed and the cabin is correctly pressurized. As for every critical system in an aircraft, it has to meet some safety regulations and it should be designed optimally in terms of weight, cost, or power consumption. This paper studies the problem of designing a DMS optimally as per the previous objectives while guaranteeing that there is redundancy, that is, the DMS is functional even if one of its components is not working. We call this new problem the DMS problem with redundancy. First, we propose a new MILP model for the DMS problem which includes redundancy, but the model turns out to be too difficult to be solved efficiently. In order to improve computational performance, a specialized branching rule and a heuristic are proposed. Computational tests are run for example instances of the DMS problem by implementing these new rules in CPLEX. It could be seen that the solving time through the new branching rule and heuristic can be significantly reduced.

Keywords: Mixed integer linear programming; Aircraft architecture; Redundancy; Branch-and-bound

1 Introduction

Redundancy and reliability are important topics in system engineering. In particular, many safety-critical systems exist in aircraft architecture, such as fly-by-wire, actuation and door management systems (DMS), that require redundancy, that is, the DMS is functional even if one of its components is not working. In this paper we propose for the first time in the literature an MILP formulation to design the DMS of an aircraft while optimizing a certain criterion (e.g., cost or weight of the system) and then we propose an algorithm to solve the model in an exact way. Going into more specific details of the problem, a DMS checks the status of doors and locks, and sends their status information to on-board computers and pressurization regulators in the aircraft. Figure 1 shows a simple DMS with one door. As every safety system, it has to adhere to safety and architectural restrictions and regulations. One of the most important regulations is that the system must be k-redundant, which means that for every task of the system (for example, checking the status of doors and locks) it has k disjoint subsystems that can fulfil the task.
As shown in Figure 2, in a usual DMS we have multiple doors. For every door, the system has to check if the door is closed and locked. Furthermore, it has to send this status information through certain units to on-board computers and also from these on-board computers to outflow valves which control the pressurisation of the aircraft. For our purposes, we will see the gathering of status information for a door and its data transfer to an on-board computer as one function of the DMS and the data transfer from the on-board computer to outflow valves as another function. These functions are implemented in the system through sets of units with cable connections between them.

*Corresponding author at: University of Edinburgh, College of Science and Engineering, School of Mathematics, James Clerk Maxwell Building, The King's Buildings, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, United Kingdom. E-mail address: lukas.schaefer@ed.ac.uk

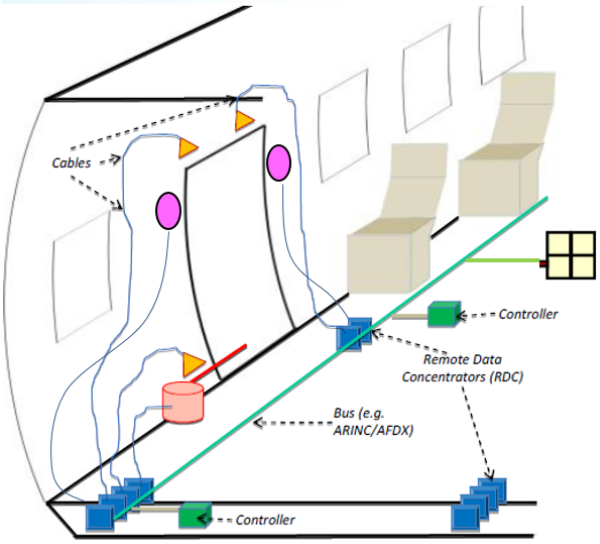

Figure 1: DMS for one door.

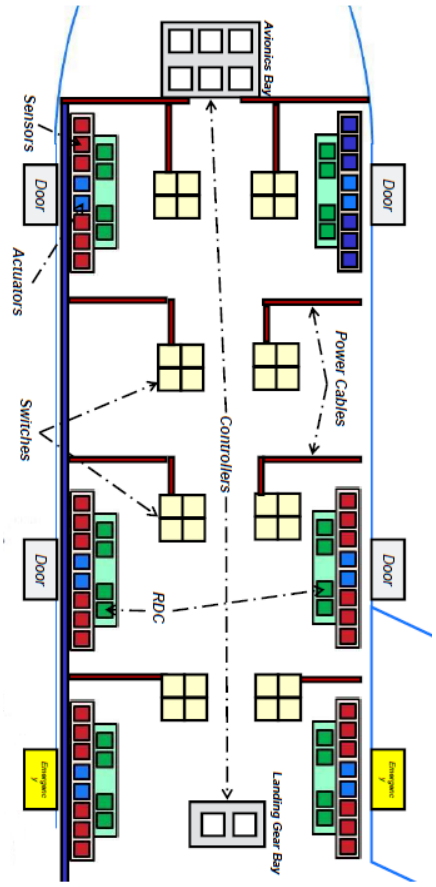

Figure 2: Example space for the DMS

Therefore, we have a DMS with multiple doors, two functions for each door and both functions have to be k-redundant per door. Our goal is to design a system meeting these constraints for which a certain objective is minimum: cost, weight, power consumption, or a weighted sum of them.
To build an MILP formulation for the DMS problem, first we have to consider one main architectural restriction. This restriction is that there are only specific locations in an aircraft where units can be installed for the DMS and also only certain cable ways are available. For our formulation, we will see these locations and possible cable connections as a directed graph $G = (N, A)$. Figure 3 shows a set of possible locations for units and its corresponding graph. Not all possible connections are shown in the figure, since otherwise the graph would be to complicated to look at.

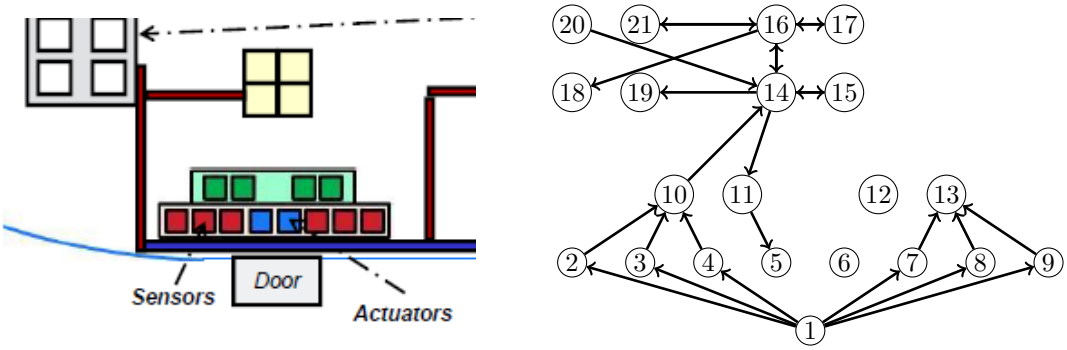

Figure 3: View of a set of locations and connections and their corresponding graph.

Therefore, the design for the DMS network is a subgraph G' of G. We will refer to the DMS network as the *overall system*. But to formulate our problem we cannot only look at the overall system. We have to look at every function which must be k-redundant in the graph. As mentioned before, functions are sets of

units and cable connections that fulfil a certain task. Hence, we can also see every function as a subgraph of G'. In the following, we refer to these subgraphs as subsystems.
Hence, k-redundancy for the functions means to have k node/arc-disjoint paths between the start and end units of the function, that is, start and end nodes are shared, but no other element is. We can see it either as node-disjoint or as arc-disjoint, because both give the same result. k node/arc-disjoint path problems have been widely researched ([9, 7, 13, 3, 4, 6]) and they have been applied in many fields, as for example routing ([10, 12]) and social networks ([5, 11]).
The problem studied in this paper is not a mere case of the previous problems. It is not a k node/arc-disjoint path problem, but it has one for every subsystem. Furthermore all subsystems are part of the overall system and do not have to be pairwise disjoint. Also, only the start and end unit of the paths are fixed, but these can be installed at different possible nodes (locations).
In this paper, we formulate the problem as an MILP and solve it using a branch-and-bound algorithm ([14]). During the computational tests, we noticed that the standard branching rules implemented in CPLEX (e.g. pseudo cost [2], strong branching [2]) do not work well and, hence, specialized branching rules need to be introduced. Neither CPLEX default heuristics work efficiently for the problem. This is a fact already seen in other papers [8]. Here, we will introduce our own specialised branching rule in which the main idea is to branch first on the overall system structure. Furthermore, we will also introduce a heuristic to find good integer solutions fast.
The rest of the paper is organized as follows: First, the MILP formulation is given in Section 2. Our specialized branching rule is discussed in Section 3 and a heuristic algorithm is proposed in Section 4. A computational study given in Section 5 shows the performance of our solution method compared to general methods for different instances of the DMS problem. The instances range from two doors to four doors with two to three paths. The performance shows that the branching rule and the heuristic that we introduce contribute to solve the problem much more efficiently. Finally, some conclusions and further perspectives are given in Section 6.

2 The DMS problem with redundancy formulation

As mentioned before, in an aircraft only certain locations are available for installing units and there is also a limited number of possible cable connections. These locations and possible connections can be seen as a directed graph $G = (N, A)$. To write the formulation concisely, we need the following notation. We first define the sets needed.

2.1 Sets

First, we need to list which unit types can be placed at which locations. Note also that a certain type of unit may come in different models: for example, a switch with 24 ports or one with 16 ports. This information is contained in the following sets:

- U, is the set of different unit types.
- $U^i,\ i \in N$, is the set of different unit types that can be put at position i.
- $N^u,\ u \in U$, is the set of different locations where a unit of type u can be set.
- $M^u,\ u \in U$, is the set of different models available for unit type u.

In Figure 4 we can see an example of these sets.

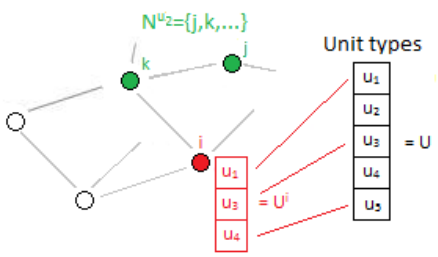

Figure 4: Example for sets U, U^i, and N_u.

Because most architectural and safety restrictions, which will be formulated as constraints, are given for the different functions, we define subsets of U, N, and A for each of these different functions. For the

DMS problem, let D be the set of doors, let F be the set of functions, and let $P = \{1, \ldots, k\}$ be the set of disjoint paths needed. We then define the following sets:

- $U_f \subset U,\ f \in F$, is the set of unit types that can be used for function f.
- $N_f = \{i \in N \mid U_f \cap U^i \neq \emptyset\},\ f \in F$, is the set of locations that can be used for function f.
- $N_f^u = \{i \in N_f \mid u \in U^i\},\ f \in F, u \in U_f$, is the set of locations that can be used for function f and unit u.
- $U_f^i = \{u \in U_f \mid u \in U^i\},\ f \in F,\ i \in N_f$, is the set of unit types that can be used at location i for function f.
- $A_f = \{(i,j) \in A \mid i,j \in N_f\},\ f \in F$, is the set of connections (arcs) that can be used for function f.
- $F^u = \{f \in F \mid u \in U_f\},\ u \in U$, is the set of functions that can use a unit of type u.

In the following, if the subscript $f \in F$ or superscript $u \in U, i \in N$ is missing, it means it is the union over the missing index. For example $U_f = \bigcup_{i \in N} U_f^i$.
There are architectural constraints that restrict which units can be connected for a function $f \in F$ or even which units must have a connection from another unit. The following sets contain this information:

- $C_f(u),\ f \in F,\ u \in U_f$, is the set of unit types to which a unit of type u can connect to for function f.
- $C_f^+(u),\ f \in F,\ u \in U_f$, is the set of unit types to which a unit of type u must connect to for function f.
- $C_f^-(u),\ f \in F,\ u \in U_f$, is the set of unit types from which a unit of type u must have a connection from for function f.
- $W_f^i(u) = U_f^i \cap C_f(u)$, is the set of unit types that can be set at location i and can have a connection from a unit of type u for function f.

Additional sets that contain information for the architectural restrictions on the functions are:

- $U_f^b,\ f \in F$, is the set of unit types that are not a start or end unit type for function f.
- $U^{s,e} = \{\ (s_f, e_f) \mid s_f,\ e_f \in U_f$ are a start unit and an end unit for some $f \in F\}$, is the set of pairs of start and end units for functions.
- $V_f^+(i) := \{k \in N_f : (i,k) \in A_f\},\ i \in N_f$, is the set of locations that location i can connect to for function $f \in F$.
- $V_f^-(i) := \{k \in N_f : (k,i) \in A_f\},\ i \in N_f$, is the set of locations that location i can connect from for function $f \in F$.
- $V_f^+(i,u) = V_f^+(i) \cap \{k \in N \mid W_f^k(u) \neq \emptyset\}$ is the set of locations that location i can connect to and where units can be placed to which unit u can be connected to.
- $V_f^-(j,\hat{u}) = V_f^-(j) \cap \{k \in N \mid \{u \in U_f^k \mid \hat{u} \in C_f(u)\} \neq \emptyset\}$ is the set of locations that location j can have a connection from and where units can be placed from which unit $\hat{u}$ can be connected from.

Lastly, we need to define the following sets, which are needed for the branching rule explained in Section 3.

- $\mathcal{U}_f^+ = \left\{u \in U_f\ :\ C_f(u) \cap \left(\bigcup_{f' \in F \setminus \{f\}} C_{f'}(u)\right) = \emptyset\right\}, f \in F$, is the set of units which cannot connect to the same units for other functions $\hat{f} \in F \setminus \{f\}$ as for $f \in F$. It is a subset of U_f.
- $\mathcal{U}_f^- = \left\{u \in U_f\ :\ \left\{\overline{u}\ :\ u \in C_f(\overline{u}) \text{ and } u \in \bigcup_{f' \in F \setminus \{f\}} C_{f'}(\overline{u})\right\} = \emptyset\right\}, f \in F$, is the set of units which cannot connect from the same units for other functions $\hat{f} \in F \setminus \{f\}$ as for $f \in F$. It is a subset of U_f.
- $E_f^+(u) = \left\{\hat{u} \in U_f \mid u \in C_f^+(\hat{u})\right\},\ f \in F,\ u \in \mathcal{U}_f^+$, is the set of units that must connect to unit u for function f.
- $E_f^-(u) = \left\{\hat{u} \in U_f \mid u \in C_f^-(\hat{u})\right\},\ f \in F,\ u \in \mathcal{U}_f^-$, is the set of units that must connect from unit u for function f.

Note that, $E_f^+(u) = \emptyset$ for $f \in F,\ u \in U_f \setminus \mathcal{U}_f^+$ and $E_f^-(u) = \emptyset$ for $f \in F,\ u \in U_f \setminus \mathcal{U}_f^-$.

2.2 Parameters

Beside the previous sets, we also need to define the following parameters which contain information about architectural restrictions on the units.

- E_{um}, number of ports available on a unit of type u and model m,
- E^{in}_{fu}, maximum number of connections that may come into a unit of type u for function f,
- E^{out}_{fu}, maximum number of connections that may go out of a unit of type u for function f,
- $T^{f,\text{out}}_{u}$, number of connections that have to leave a unit of type u. Note that this value is equal to $|C^{+}_{f}(u)|$,
- $T^{f,\text{in}}_{u}$, number of connections that have to come into a unit of type u. Note that this value is equal to $|C^{-}_{f}(u)|$.

2.3 Decision variables

We will now define the decision variables. First, we need variables that represent the overall system. These are assignment variables that represent which unit model is installed at which location and variables that represent which cable connections are used.

- $t_{ium} \in \{0,1\},\ i \in N,\ u \in U^i,\ m \in M^u$, is a binary variable that takes value 1 if unit type u and model m is set at location i and 0 otherwise.
- $x_{ij} \in \{0,1\},\ (i,j) \in A$, is a binary variable that takes value 1 if locations i and j are connected, that is, if arc (i,j) is used.
- $x_{iju\hat{u}} \in \{0,1\},\ (i,j) \in A, u \in U^i,\ \hat{u} \in W^j(u)$, is a binary variable that takes value 1 if locations i and j are connected and units u and $\hat{u}$ are installed at i and j, respectively. It is 0 otherwise.

We need the same kind of variables for every path in a subsystem. Let $(d,f,p) \in D \times F \times P$ refer to path p of the subsystem for function f and door d. The required decision variables for every $(d,f,p) \in D \times F \times P$ are:

- $t^{dfp}_{iu} \in \{0,1\},\ i \in N_f,\ u \in U^i_f$, a binary variable that takes value 1 if a unit of type u is installed at location i, and
- $x^{dfp}_{iju\hat{u}} \in \{0,1\},\ (i,j) \in A_f,\ u \in U^i_f,\ \hat{u} \in W^j_f(u)$, a binary variable that takes value 1 if locations i and j are connected and unit u and $\hat{u}$ are installed at i and j, respectively. It is 0 otherwise.

These variables are enough to write our model. But for the new branching rule and pruning rule, we also need the following decision variables:

- $t_u \in \{0,1,\ldots,|N^u|\},\ u \in U^i$, which is the total number of units of type u used,
- $t_{um} \in \{0,1,\ldots,|N^u|\},\ u \in U^i,\ m \in M^u$, which is the total number of units of type u and model m used,
- $t^f_u \in \{0,1,\ldots,|N^u|\},\ f \in F,\ u \in U^i_f$, which is the total number of units of type u used for function f,
- $t^f_{iu} \in \{0,1\},\ f \in F,\ u \in U^i_f$, which takes value 1 if a unit of type u is used for function f at location i, and 0 otherwise.

2.4 Constraints

Here we describe the constraints of the model. With the overall system and the subsystems, we have two levels of decision variables which have to be synchronized. For example, if a unit or cable connection is installed in the overall system, there has to be a subsystem which uses it. And also the other way around, if a unit or cable connection is used for a subsystem, it has to be installed in the overall system. Constraints (1) to (4) force this synchronization. Furthermore, constraints (2) and (4) also impose that the subsystems have k node/arc-disjoint paths. Constraint (1) forces at least one subsystem to use a unit u at location i if it is located there in the overall system.

$\forall i \in N,\ \forall u \in U^i$:

$$\sum_{m \in M^u} t_{ium} \leq \sum_{\substack{d \in D\ p \in P,\\ f \in F(u)}} t^{dfp}_{iu}, \tag{1}$$

Constraint (2) forces at least one subsystem to use a connection (i,j) if it is used in the overall system.

$\forall (i,j) \in A$:

$$x_{ij} \leq \sum_{\substack{d \in D,\ p \in P,\\ f \in F \ \mid \ (i,j) \in A_f,\\ u \in U_f^i,\ \hat{u} \in W_f^j(u)}} x_{iju\hat{u}}^{dfp}, \tag{2}$$

Constraint (3) forces the overall systems to locate a unit u at location i if it was used in at least one subsystem. Furthermore, it imposes that the subsystems have k node-disjoint paths.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i\ ,\ \forall \hat{f} \in F^u$:

$$\sum_{m \in M^u} t_{ium} \geq t_{iu}^{dfp} + \sum_{\hat{p} \in P \setminus \{p\}} t_{iu}^{d\hat{p}\hat{f}}, \tag{3}$$

Constraint (4) forces the overall systems to use connection (i,j) if it was used in at least one subsystem. Furthermore, it imposes that the subsystems have k arc-disjoint paths.

$\forall d \in D,\ \forall f \in F,\ \forall (i,j) \in A_f,\ \forall u \in U^i,\ \forall \hat{u} \in W^j(u)$:

$$x_{iju\hat{u}} \geq \sum_{p \in P} x_{iju\hat{u}}^{dfp}. \tag{4}$$

Through these constraints (1) to (4) , the binary requirement on t_{ium} and $x_{iju\hat{u}}$ variables can be lifted and changed to be continuous in $[0,1]$.
We also have to synchronize the decision variables on the same level. For example, in the overall system we have the decision variables $x_{iju\hat{u}}$ and x_{ij}, both of which have the information on whether the cable connection between locations i and j is used. Constraints (5) to (7) synchronize the decision variables of the overall system.

$\forall (i,j) \in A$:

$$\sum_{\substack{u \in U^i,\\ \hat{u} \in W^j(u)}} x_{iju\hat{u}} = x_{ij}, \tag{5}$$

$\forall (i,j) \in A,\ \forall u \in U^i$:

$$\sum_{\hat{u} \in W^j(u)} x_{iju\hat{u}} \leq \sum_{m \in M^u} t_{ium}, \tag{6}$$

$\forall (i,j) \in A,\ \forall \hat{u} \in U^j$:

$$\sum_{u \in U^i \ \mid \ \hat{u} \in C(u)} x_{iju\hat{u}} \leq \sum_{m \in M^u} t_{j\hat{u}m}. \tag{7}$$

Constraints (8) to (11) are the flow constraints for the different subsystems. Constraint (8) imposes that no more than the maximum number of connections E_{fu}^{out} can go out of unit u at location i for a function $f \in F$ in a subsystem and synchronizes the arc and location variables of the subsystems.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ i \in N_f,\ \forall u \in U_f^i$:

$$\sum_{\substack{j \in V_f^+(i,u),\\ \hat{u} \in W_f^j(u)}} x_{iju\hat{u}}^{dfp} \leq E_{fu}^{out} t_{iu}^{dfp}, \tag{8}$$

Constraint (9) imposes the same as (8), but for connections arriving at a location.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ j \in N_f,\ \forall \hat{u} \in U_f^j$:

$$\sum_{\substack{i \in V_f^-(j,\hat{u}),\\ u \in U_f \mid \hat{u} \in C_f(u)}} x_{iju\hat{u}}^{dfp} \leq E_{fu}^{in} t_{j\hat{u}}^{dfp}. \tag{9}$$

Constraint (10) imposes that the flow arrives. If a unit u at location i is used for a subsystem, at least $T_u^{f,in}$ connections have to arrive at location i for the subsystem. Constraint (11) imposes that the flow continues. This means that if a unit u at location i is used for a subsystem, at least $T_u^{f,out}$ connections have to leave location i for the subsystem.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F, \forall i \in N_f$:

$$\sum_{u \in (U_f^{\mathrm{b}} \cup \{e_f\}) \cap U_f^i} T_u^{f,\mathrm{in}} t_{iu}^{dfp} \leq \sum_{\substack{\ell \in V_f^-(i),\ u \in (U_f^{\mathrm{b}} \cup \{e_f\}) \cap U_f^i,\\ \hat{u} \in U_f \mid u \in C_f(\hat{u})}} x_{\ell i\hat{u}u}^{dfp}, \tag{10}$$

$$\sum_{u \in (U_f^{\mathrm{b}} \cup \{s_f\}) \cap U_f^i} T_u^{f,\mathrm{out}} t_{iu}^{dfp} \leq \sum_{\substack{\ell \in V_f^+(i),\ u \in (U_f^{\mathrm{b}} \cup \{s_f\}) \cap U_f^i,\\ \hat{u} \in W_f^{\ell}(u)}} x_{i\ell u\hat{u}}^{dfp}. \tag{11}$$

Constraint (12) makes sure that the number of available ports for connections is not exceeded.

$\forall i \in N,\ \forall u \in U^i$:

$$\sum_{\substack{\ell \in V^-(i,u),\\ \hat{u} \in U^{\ell} \mid u \in C(\hat{u})}} x_{\ell i\hat{u}u} + \sum_{\substack{\ell \in V^+(i,u),\\ \hat{u} \in W^k(u)}} x_{i\ell u\hat{u}} \leq \sum_{m \in M^u} t_{ium} E_{um}. \tag{12}$$

To have k paths in every subsystem, constraint (13) imposes that a start unit and an end unit are installed for all k paths for every function.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall u_f \in \{s_f, e_f\}$:

$$\sum_{i \in N_f^{u_f}} t_{iu_f}^{dfp} = 1. \tag{13}$$

In some cases, specific functions have to be connected. This means that, for example, the end unit of a function f_1 has to be also the start unit of another function f_2. Therefore, we need for some $f_1, f_2 \in F$ that

$\forall i \in N^{e_{f_1}}, \forall d \in D,\ \forall p \in P$ with $e_{f_1} = s_{f_2}$:

$$t_{ie_{f_1}}^{df_1p} = t_{is_{f_2}}^{df_2p}. \tag{14}$$

Further architectural restrictions are given by (15) and (16). Constraint (15) means that if a unit u is installed, then it connects to all unit types it has to connect to.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i,\ \forall \hat{u} \in C_f^+(u)$:

$$t_{iu}^{dfp} \leq \sum_{j \in V_f^+(i) \cap N_f^{\hat{u}}} x_{iju\hat{u}}^{dfp}. \tag{15}$$

Constraint (16) means that if a unit u is installed, then it has a connection from all unit types it has to have a connection from.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall j \in N_f,\ \forall u \in U_f^j,\ \forall \hat{u} \in C_f^-(u)$:

$$t_{ju}^{dfp} \leq \sum_{i \in V_f^-(j) \cap N_f^{\hat{u}}} x_{ij\hat{u}u}^{dfp}. \tag{16}$$

Finally, we have that at most one unit is installed at any location.

$\forall i \in N$:

$$\sum_{\substack{u \in U^i, \\ m \in M^u}} t_{ium} \leq 1. \tag{17}$$

Until now, we gave an MILP formulation for a general network system with multiple sets of functions and k-redundancy for functions. The following constraint is specific for the DMS problem with redundancy and is a safety requirement. In the DMS we always have two unit types named outflow valve (OVF) and outflow valve control unit (OCU). The requirement is that every OVF must have at least a connection from two different OCUs in the overall system. This is imposed through constraint (18):

$\forall i \in N^{OVF}$:

$$\sum_{m \in M^{OVF}} x_{i(OVF)m} \leq \sum_{j \in N^{OCU} \cap V^-(i)} 2x_{ji(OCU)(OVF)}. \tag{18}$$

2.5 Tightening constraints

With constraints (1)-(17) we have a valid formulation for the DMS problem with redundancy. Furthermore, we could reduce the number of decision variables constraints by aggregating the decision variables $x_{iju\hat{u}}$ to x_{ij} and still have a valid MILP model for the DMS problem. But we noted during our research that this model was not useful as we encountered large gaps between the LP relaxation and the MILP solution which lead to long computational times. Therefore, this model is not included in this paper.
The following constraints (19)-(22) strengthen the previous model (1)-(17), but they are not necessary for the DMS problem to be well modeled. Constraints (19) and (20) strengthen the flow of the different paths in the subsystems through sets $C_f^+(u)$ and $C_f^-(u)$ and the parameters E_{fu}^{in} and E_{fu}^{out}. $C_f^+(u)$ and $C_f^-(u)$ are the sets that contain information on which unit must connect to which unit and which unit must connect from which other unit,respectively. Furthermore, E_{fu}^{in} and E_{fu}^{out} are the parameters that contain the information what the maximum number of connections arriving or leaving a unit u is for function $f \in F$.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall u \in U_f$:

$$\sum_{\substack{\hat{u} \in \{\overline{u} \ \mid \ u \in C_f^+(\overline{u})\}, \\ i \in N_f^{\hat{u}}}} t_{i\hat{u}}^{dfp} \leq E_{fu}^{in} \sum_{i \in N_f^u} t_{iu}^{dfp} \tag{19}$$

$$\sum_{\substack{\hat{u} \in \{\overline{u} \ \mid \ u \in C_f^-(\overline{u})\}, \\ i \in N_f^{\hat{u}}}} t_{i\hat{u}}^{dfp} \leq E_{fu}^{out} \sum_{i \in N_f^u} t_{iu}^{dfp} \tag{20}$$

Constraints (21) and (22) synchronize decision variable sets t_{ium} and x_{ij}.

$\forall i \in N\ \ \forall u \in (U^{\mathrm{b}} \cup \bigcup_{f \in F} \{e_f\}) \cap U^i$:

$$\sum_{m \in M^u} t_{ium} \leq \sum_{\ell \in V^-(i,u)} x_{\ell i} \tag{21}$$

$\forall i \in N\ \ \forall u \in (U^{\mathrm{b}} \cup \bigcup_{f \in F} \{s_f\}) \cap U^i$:

$$\sum_{m \in M^u} t_{ium} \leq \sum_{\ell \in V^+(i,u)} x_{i\ell}. \tag{22}$$

The utility of constraints (19)-(22) was tested and the results can be seen in Table 2 in Section 5. It can be clearly seen that the additional constraints reduce the gap significantly and help to solve the problem faster.

2.6 Synchronization constraints for branching variables

The following constraints impose the synchronization between decision variables t_{ium} and t_{iu}^{dfp} with the branching variables $t_u,\ t_{um},\ t_u^f,$ and t_{iu}^f.

$\forall u \in U,\ \forall m \in M^u$:

$$\sum_{i \in N^u} t_{ium} = t_{um}, \tag{23}$$

$\forall u \in U$:

$$\sum_{m \in M^u} t_{um} = t_u, \tag{24}$$

$\forall d \in D,\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i$:

$$\sum_{p \in P} t_{dpfiu} \leq t_{iu}^f, \tag{25}$$

$\forall f \in F,\ \forall u \in U_f$:

$$\sum_{i \in N_f} t_{iu}^f = t_u^f. \tag{26}$$

2.7 Objective functions

We introduce here several possible objective functions. The following parameters are used:

- c_{um}: cost of a unit of type $u \in U$ and model $m \in M^U$ in Euro,
- c_0: cost of cable per meter in Euro,
- w_{um}: weight of a unit of type $u \in U$ and model $m \in M^U$ in grams,
- w_0: weight of cable per meter in grams,
- pow_{um}: power usage of a unit of type $u \in U$ and model $m \in M^U$ in kWh,
- ℓ_{ij}: distance between position i and j in meters,
- $\gamma_c,\ \gamma_w,\ \gamma_{pow}$: weight for the respective parameter in a combination of different objectives.

The different objective functions that we consider for minimization are the following:

1. Total of weight and cost (obj1):

$$\min \left[\sum_{i \in N} \sum_{\substack{u \in U^i, \\ m \in M^u}} t_{ium} \left(c_{um} + w_{um} \right) \right] + \sum_{(i,j) \in A} x_{ij} \ell_{ij} \left(c_0 + w_0 \right), \tag{27}$$

2. Total number of units placed (obj2):

$$\min \left[\sum_{i \in N} \sum_{\substack{u \in U^i, \\ m \in M^u}} t_{ium} \right], \tag{28}$$

3. Total length of cable used (obj3):

$$\min \sum_{(i,j) \in A} x_{ij} l_{ij}, \tag{29}$$

4. Weighted total of weight, cost, and power (obj4):

$$\min \left[\sum_{i \in N} \sum_{\substack{u \in U^i, \\ m \in M^u}} t_{ium} \left(\gamma_c c_{um} + \gamma_w w_{um} + \gamma_{pow} pow_{um} \right) \right] + \sum_{(i,j) \in A} x_{ij} \ell_{ij} \left(\gamma_c c_0 + \gamma_w w_0 \right). \tag{30}$$

3 Branching rule

In this section we introduce a problem specific branching rule that will allow us to solve our problem much more efficiently. In the literature, see [2], branching is usually done on decision variables which must have an integer value but which have a fractional value for the linear relaxation of the current node. Computational tests have shown that this strategy is not efficient for the DMS problem. Therefore in the new branching rule that we propose here:

- Whenever possible, we branch on the decision variables chosen among $t_u,\ t_{um}$ and t_u^f.
- We are allowed to choose them even if they have an integer value for the linear relaxation of the node that is being considered.
- If no suitable decision variable is found among $t_u,\ t_{um}$ and t_u^f, general branching rules of CPLEX are used.

In the following, all variables with a star $*$, e.g. t_{um}^*, are the values of those variables for the linear relaxation at the current node.
The new branching rule branches first on the decision variables t_u and t_{um} to have t_u fixed to a value and to have all t_{um} integer feasible.The objective functions are all mainly based on cost, weight or power usage of installed units and cables in the overall system and the number of cables is strongly linked to the number of installed units. Because of this and by having the variables t_u fixed and t_{um} integer feasible, we have small gaps between lower and upper bound of the objective value in the branch-and-bound tree. With the smaller gaps and by using a efficient heuristic, pruning becomes more efficient during branching. Afterwards it branches on the decision variables t_f^u to have them equal to their lower bounds $t_u^{f,lb}$ in the linear relaxation of the nodes in the branching tree. The decision variables t_f^u have to be equal to their lower bound to decide if it is necessary to branch on integer feasible decision variables t_{um} or not.
In the solution of a linear relaxation, there is no guarantee that the redundancy requirement will be met. For example, the solution of a linear relaxation can have

$$t_{iu}^{dfp,*} = t_{iu}^{df\hat{p},*} = 0.5$$

with $d \in D,\ f \in F,\ i \in N_f, u \in U_f^i,\ p, \hat{p} \in P$ and $p \neq \hat{p}$. Because of this, a node may have a combination of units given by t_u and t_{ium} that do not provide enough ports for every unit type. A small example is given in Appendix A. It is very hard to prune such nodes through normal branching and no cuts where found to prevent such nodes from occurring. Therefore, by having the decision variables t_u fixed, t_f^u equal to its lower bound and t_{ium} integer feasible, we can test for a unit type u, whose ports are all used in the linear relaxation, if the combination of units at the current node can have a feasible integer solution. We will refer to this test as *validity test*. A node is valid if it can provide enough ports at every unit to obtain an integer feasible solution.
Through the previous branching, we have obtained that $t_u^{ub} = t_u^{lb}$ for all $u \in U$, and that $t_u^{f,*} = t_u^{f,lb}$ for all $f \in F$. To test the validity of a unit u and the current combination of units installed, we need the sets $E_f^+(u)$ and $E_f^-(u)$ which were defined in Section 2.1.
Let $\overline{u}$ be the unit type for which we do the validity test. Because $t_{\overline{u}}^{ub} = t_{\overline{u}}^{lb}$, $t_{\overline{u}}^{f,*} = t_{\overline{u}}^{f,lb}$ for all $f \in F$ and constraints (15) and (16) hold, we know that at least $\sum_{f \in F} \sum_{\hat{u} \in E_f^+(\overline{u})} t_{\hat{u}}^{f,lb}$ connections have to arrive through and $\sum_{f \in F} \sum_{\hat{u} \in E_f^-(\overline{u})} t_{\hat{u}}^{f,lb}$ have to leave the units u installed in the overall system. Through the values $t_{\overline{u}}^*$, $m \in M^U$, we can build all possible combinations of connections that can arrive at the units $\overline{u}$ and leave from $\overline{u}$ for the different functions and check if they are feasible. If it is shown that not one combination is feasible, then the unit type failed the validity test and we branch on a variable $t_{\overline{u}m}$ with $t_{\overline{u}m} \leq t_{\overline{u}m}^* - 1$ and $t_{\overline{u}m} \geq t_{\overline{u}m}^* + 1$. This does not cut off any valid integer feasible solution. The validity test is made for all unit types u whose ports are all used in the linear relaxation.
If no $u \in U$ fails the validity test, we do not branch on integer feasible t_{um} variables, and standard branching rules implemented in the solver are used. The branching rule is summarized in Algorithm 1.

Algorithm 1 Branching rule

Require: LP Relaxation (t^*, x^*)
  for $u \in U$ **do**
    if t_u^* integer infeasible **then**
      Branch on t_u with $t_u \geq \lceil t_u^* \rceil$ and $t_u \leq \lfloor t_u^* \rfloor$
    else if t_{um}^* integer infeasible **then**
      Branch on t_{um} with $t_{um} \geq \lceil t_{um}^* \rceil$ and $t_{um} \leq \lfloor t_{um}^* \rfloor$
    else if $t_u^* \neq t_u^{lb}$ **then**
      Branch on t_u with $t_u \geq t_u^*$ and $t_u \leq t_u^* - 1$
    else if $t_u^* \neq t_u^{ub}$ **then**
      Branch on t_u with $t_u \geq t_u^* + 1$ and $t_u \leq t_u^*$
    else if $t_u^{f,*} \neq t_u^{f,lb}$ **then**
      Branch on t_u^f with $t_u^f \geq \lceil t_u^{f,*} \rceil$ and $t_u^f \leq \lceil t_u^{f,*} \rceil - 1$
    end if
  end for
  for $u \in U$ **do**
    if u fails validity test **then**
      Choose $m \in M^U$ with $t_{um}^* \geq 1$ and branch with $t_{um} \leq t_{um}^* - 1$ and $t_{um} \geq t_{um}^* + 1$, while remembering that $t_u m$ is integer feasible.
    end if
  end for
  if No decision variable was chosen for branching **then**
    Use CPLEX standard branching rules
  end if

4 Heuristic

The formulation for the DMS problem has many binary variables. Furthermore, in most of the constraints in the model which involve sums of binary variables, most of the variables have values much smaller than 0.5. In most general heuristics, e.g. feasibility pump [1], we cannot deal efficiently with such a problem and they result in long heuristic times and poor feasible solutions.
Therefore, we propose the following heuristic whose aim is to provide fast good feasible solutions, which will help us to solve the DMS problem in an exact way much more efficiently. The idea of the algorithm is to start with a path of a subsystem and to find an integer feasible solution for it. Afterwards we move to another path of a subsystem. After all paths are made integer feasible, we have also an integer feasible overall system.
Let $S = \{(d, f, p) \in D \times F \times P\}$ be the set of all paths. To decide with which path to start, we calculate

$$v_{dfp} = \left(\sum_{x_{ijuû}^{dfp} \neq 0} \left(1 - x_{ijuû}^{dfp,*}\right) \cdot \sum_{x_{ijuû}^{dfp} \neq 0} 1 \right) \tag{31}$$

for all $(d, f, p) \in S$ where $-x_{ijuû}^{dfp,*}$ are the variable values of the linear relaxation. We call v_{dfp} *violation of the path.* Let $(\hat{d}, \hat{f}, \hat{p})$ be the path with the smallest violation. To obtain an integer feasible solution for decision variables of $(\hat{d}, \hat{f}, \hat{p})$, we look at the connection variables $x_{ijuû}^{\hat{d}\hat{f}\hat{p}}$. We define $1 - x_{ijuû}^{\hat{d}\hat{f}\hat{p}}$ as the violation of the variable. Now we choose the least violated variable and set it to 1. Afterwards, we solve the new problem and choose the new least violated variable. If all connection variables of the path are binary feasible, which means that they have value 0 or 1, then all location variables of this path are binary feasible because of the synchronization constraints. Now that this path is integer feasible, we choose the new least violated path and repeat this procedure until all the paths are feasible for the integer problem. When every path is feasible, we have feasible subsystems, but the overall system may still have infeasible variables. Constraint 3 synchronizes the decision variables of the subsystems with the overall system. But because on the left hand side we have a sum over the models of the unit, the decision variables still t_{ium} of the overall system still can be fractional in the linear relaxation even though all subsystem variables are integer feasible. Therefore, if we have several variables $t_{ium} > 0$ for a unit u and location i, we choose the model $m \in M^U$ with the most ports E_{um} for unit u, set t_{ium} to 1 and solve the linear relaxation. This is repeated until all variables t_{ium} are integer feasible. By proceeding in this way we obtain a feasible solution that in most cases is better than the solutions obtained by standard heuristics implemented in

CPLEX. Particularly, the greater the cardinality of D, F or P is, the better our heuristic works when compared to CPLEX heuristics. Our heuristic is shown in pseudo-code in Algorithm 2.

Algorithm 2 Model Heuristic

Require: LP Relaxation (t^*, x^*), K = number of violated variables,
K_f = number of violated variables on flow level f, $S = \{(d, f, p) \in D \times F \times P\}$
while $K_f \neq 0$ **do**
$v_{dfp} = \left(\sum_{x_{iu\hat{u}}^{dfp} \neq 0} \left(1 - x_{iju\hat{u}}^{dfp,*}\right) \cdot \sum_{x_{iu\hat{u}}^{dfp} \neq 0} 1\right)$
$(d, f, p)^* = \arg\min_{(d,f,p) \in S}\{v_{dfp}\}$
$S = S \backslash \{(d, f, p)^*\}$
$K_{(d,f,p)^*}$ = number of violated variables in the $(d, f, p)^*$ subsystem
Let $V_{(d,f,p)^*}$ be the set of all $x_{iju\hat{u}}^{(dfp)^*}$ variables in subsystem $(d, f, p)^*$
while $K_{(d,f,p)^*} \neq 0$ **do**
$\hat{x} = \arg\min\{x_{iju\hat{u}}^{(dfp)^*,*} - 1\}$
Set $\hat{x} = 1$ and solve LP
$V_{(d,f,p)^*} = V_{(d,f,p)^*} \backslash \{\hat{x}\}$
end while
end while
while $K \neq 0$ **do**
$(\hat{i}, \hat{u}) = \arg\min_{i \in N, u \in U, m \in M^u}\{|t_{ium} - 0.5|\}$
$\hat{m} = \arg\max_{m \in M^{\hat{u}}}\{E_{\hat{u}m}\}$
Set $t_{\hat{i}\hat{u}\hat{m}} = 1$ and solve LP
end while

5 Computational Results

We carry out a computational study to analyse the performance of our new branching rule and heuristic. We run the same instances using CPLEX with and without them. The problem instances differ in the number of doors, number of disjoint paths required, configurations of the locations and objective functions. The instances are artificial but they were built based on data provided by AIRBUS Group. As mentioned in the introduction, we have two functions in the DMS and therefore $F = \{F1,\ F2\}$. The first function is the information flow from two sensors at a door to a controller (CPIOM) and the second function is the information flow from that controller to an outflow valve. Also, the controller used in both functions has to be the same for a door and therefore constraint (14) is used.
The DMS has 8 unit types: door (DO), latch &lock sensor (LLS), closed sensor (CS), outflow valve (OVF), outflow valve control unit (OCU), remote data concentrator (RDC), CPIOM, switch (SWT). We also have the following unit sets:

- $U^{s,e} = \{s_1 = DO, s_2 = CPIOM, e_1 = CPIOM, e_2 = OVF\}$,
- $U^{\mathrm{b}} = \{LLS, CS, OCU, RDC, SWT\}$,
- $U_1 = \{DO, LLS, CS, RDC, CPIOM, SWT\}$,
- $U_2 = \{OVG, OCU, RDC, CPIOM, SWT\}$.

Except for the door, all units are available in different models . For example, there can be various kinds of RDC models with different costs, weights or number of ports.
In Table 1, the size of the model for different instances is shown. It includes the number of potential locations, variables, integer variables, binary variables, and constraints. For the two door instances, a more complex configuration of locations was used which resulted in a higher number of continuous variables. The more complex configuration of locations was not used for more doors to avoid having too large problems which were impossible to solve.

Doors	Locations	Continuous Variables	Binary Variables	Integer Variables	Constraints
2	66	3149	13038	21	14020
3	90	1772	21648	21	23468
4	91	1820	27226	21	29262

Table 1: Size of the problems for different number of doors and locations.

In the computational tests, a total of 5 different cost and weight sets were used and a time limit of 4 hours was set. All computations were done with a Four Intel Xeon E5-2680 v3 2.5GHz, 192Gb RAM and CPLEX 12.5.1. Furthermore, implementation of the branching rule and heuristic where done in C++. Also, all available cuts in CPLEX were disabled because computational tests showed that they are not efficient for this problem. The problem instances in the tables have the names $i(d, p, loc, u, \ell)$ with d the number of doors, p the number of paths, loc the number of locations in the configuration of the space, and ℓ the number of the cost and weight set.
In Table 2, we solve the DMS problem with redundancy formulation with and without constraints (19)-(22). Column "Objective Function" refers to which objective function was used, z_{MIP} is the MIP optimal value of the instance, z_{LP} is the optimal value for the linear relaxation at the root node, Gap is the gap at the root node and Time is the solution time in seconds.

Instance	Objective Function	z_{MIP}	Without			With		
			z_{LP}	Gap	Time	z_{LP}	Gap	Time
$i(2,2,66,1)$	obj1	2721.92	2608.06	4.18	14400	2721.92	0.00	20
$i(2,2,66,1)$	obj2	22.00	17.00	22.73	14400	22 .00	0.00	19
$i(2,2,66,1)$	obj3	32.00	26.00	18.75	14400	32 .00	0.00	22
$i(2,2,66,1)$	obj4	4127.88	3847.19	6.80	14400	4127.88	0.00	16
$i(2,2,66,2)$	obj1	2539.91	2473.28	2.62	14400	2539.91	0.00	13
$i(2,2,66,3)$	obj1	2257.46	2173.33	3.73	14400	2257.46	0.00	11
$i(2,2,66,4)$	obj1	2871.12	2728.16	4.98	14400	2871.12	0.00	13
$i(2,2,66,5)$	obj1	3721.74	3592.35	3.48	14400	3721.74	0.00	17
$i(2,3,66,1)$	obj1	4082.88	3912.09	4.18	14400	4082.88	0.00	98
$i(2,3,66,1)$	obj2	32.00	24.50	23.44	14400	32.00	0.00	177
$i(2,3,66,1)$	obj3	48.00	39.00	18.75	14400	48.00	0.00	94
$i(2,3,66,1)$	obj4	6191.82	5770.78	6.80	14400	6191.82	0.00	110
$i(2,3,66,2)$	obj1	3809.86	3709.93	2.62	14400	3809.86	0.00	290
$i(2,3,66,3)$	obj1	3386.19	3260.00	3.73	14400	3386.19	0.00	144
$i(2,3,66,4)$	obj1	4306.68	4092.24	4.98	14400	4306.68	0.00	167
$i(2,3,66,5)$	obj1	5582.60	5388.54	3.48	14400	5582.60	0.00	197
$i(3,2,90,1)$	obj1	2731.00	2608.50	4.49	14400	2731.00	0.00	57
$i(3,2,90,1)$	obj2	27.00	18.00	33.33	14400	27.00	0.00	72
$i(3,2,90,1)$	obj3	4.000	30.00	25.00	14400	40.00	0.00	164
$i(3,2,90,1)$	obj4	4182.22	3854.49	7.84	14400	4182.22	0.00	148
$i(3,2,90,2)$	obj1	2545.63	2473.28	2.84	14400	2545.63	0.00	252
$i(3,2,90,3)$	obj1	2262.84	2173.33	3.96	14400	2262.84	0.00	117
$i(3,2,90,4)$	obj1	2876.44	2728.16	5.15	14400	2876.44	0.00	355
$i(3,2,90,5)$	obj1	3731.41	3592.36	3.73	14400	3731.41	0.00	183
$i(3,3,90,1)$	obj1	4096.50	3912.75	4.49	14400	4096.50	0.00	758
$i(3,3,90,1)$	obj2	39.00	25.50	34.62	14400	39.00	0.00	2248
$i(3,3,90,1)$	obj3	60.00	45.00	25.00	14400	60.00	0.00	1952
$i(3,3,90,1)$	obj4	6273.33	5781.73	7.84	14400	6273.33	0.00	3486
$i(3,3,90,2)$	obj1	3818.45	3709.93	2.84	14400	3818.45	0.00	4437
$i(3,3,90,3)$	obj1	3394.27	3260.00	3.96	14400	3394.27	0.00	1792
$i(3,3,90,4)$	obj1	4314.66	4092.24	5.15	14400	4314.66	0.00	1218
$i(3,3,90,5)$	obj1	5597.12	5388.54	3.73	14400	5597.12	0.00	3008
$i^*(3,2,90,1)$	obj1	2829.00	2608.50	7.79	14400	2731 .00	3.46	14400

$i^*(3,2,90,1)$	obj2	27.00	18.00	33.33	14400	27.00	0.00	62
$i^*(3,2,90,1)$	obj3	40.00	30.00	25.00	14400	40.00	0.00	59
$i^*(3,2,90,1)$	obj4	4182.22	3854.49	7.84	14400	4182.22	0.00	144
$i^*(3,2,90,2)$	obj1	2644.33	2473.28	6.47	14400	2545.63	3.73	14400
$i^*(3,2,90,3)$	obj1	2336.34	2173.33	6.98	14400	2262.84	3.15	14400
$i^*(3,2,90,4)$	obj1	2886.24	2728.16	5.48	14400	2876.44	0.34	14400
$i^*(3,2,90,5)$	obj1	3869.73	3592.36	7.17	14400	3731.41	3.57	14400
$i(4,2,91,1)$	obj1	2838.08	2626.22	7.46	14400	2740.08	3.45	14400
$i(4,2,91,1)$	obj2	32.00	27.00	15.63	14400	32.00	0.00	752
$i(4,2,91,1)$	obj3	48.00	42.00	12.50	14400	48.00	0.00	855
$i(4,2,91,1)$	obj4	4236.56	3955.87	6.63	14400	4236.56	0.00	65
$i(4,2,91,2)$	obj1	2650.06	2484.73	6.24	14400	2551.36	3.72	14400
$i(4,2,91,3)$	obj1	2341.73	2184.09	6.73	14400	2268.23	3.14	14400
$i(4,2,91,4)$	obj1	2891.56	2738.80	5.28	14400	2881.76	0.34	14400
$i(4,2,91,5)$	obj1	3879.41	3611.71	6.90	14400	3741.09	3.57	14400
Average Computing times					14400			3492

Table 2: Analysis of constraints (19)- (22)

It is evident that the use of constraints (19)- (22) helps considerably to solve the DMS problem formulation. In most of the cases, we were able to find an optimal solution at the root node whereas, when we did not use them, we were not able to solve the problem in 4 hours. Nevertheless, there are some instances which we cannot solve within the time limit even with the use of these constraints. It must be noted that for all instances that could not be solved to optimality, the MILP optimal solution was found as best upper bound but the gap could not be closed to prove optimality and the lower bound did not improve after the root node.

In the following, we compare solving the problem with standard solver CPLEX, referred to as "Standard" in Table 3 and CPLEX with the new branching rule and heuristic, referred to as "Proposed". In both cases, as mentioned earlier, the solver cuts were disabled. To compare the performance, Table 3 shows the different instances, objective function used, linear relaxation value (z_{LP}) at the root node, gap in % and the times, respectively. Independently on whether optimality is achieved or not, z_{lb} and z_{ub} show the best lower bound and the best upper bound when the run stops. If z_{ub} is in bold face, it means that the optimal solution was found.

Instance	Objective Function	z_{LP}	Gap	Standard			Proposed		
				Time	z_{lb}	z_{ub}	Time	z_{lb}	z_{ub}
$i(2,2,66,1)$	obj1	2721.92	0.00	20	2721.92	**2721.92**	22	2721.92	**2721.92**
$i(2,2,66,1)$	obj2	22.00	0.00	19	22.00	**22.00**	24	22.00	**22.00**
$i(2,2,66,1)$	obj3	32.00	0.00	22	32.00	**32.00**	20	32.00	**32.00**
$i(2,2,66,1)$	obj4	4127.88	0.00	16	4127.88	**4127.88**	22	4127.88	**4127.88**
$i(2,2,66,2)$	obj1	2539.91	0.00	13	2539.91	**2539.91**	29	2539.91	**2539.91**
$i(2,2,66,3)$	obj1	2257.46	0.00	11	2257.46	**2257.46**	29	2257.46	**2257.46**
$i(2,2,66,4)$	obj1	2871.12	0.00	13	2871.12	**2871.12**	27	2871.12	**2871.12**
$i(2,2,66,5)$	obj1	3721.74	0.00	17	3721.74	**3721.74**	27	3721.74	**3721.74**
$i(2,3,66,1)$	obj1	4082.88	0.00	98	4082.88	**4082.88**	69	4082.88	**4082.88**
$i(2,3,66,1)$	obj2	32.00	0.00	177	32.00	**32.00**	99	32.00	**32.00**
$i(2,3,66,1)$	obj3	48.00	0.00	94	48.00	**48.00**	94	48.00	**48.00**
$i(2,3,66,1)$	obj4	6191.82	0.00	110	6191.82	**6191.82**	66	6191.82	**6191.82**
$i(2,3,66,2)$	obj1	3809.86	0.00	290	3809.86	**3809.86**	91	3809.86	**3809.86**
$i(2,3,66,3)$	obj1	3386.19	0.00	144	3386.19	**3386.19**	223	3386.19	**3386.19**
$i(2,3,66,4)$	obj1	4306.68	0.00	167	4306.68	**4306.68**	72	4306.68	**4306.68**
$i(2,3,66,5)$	obj1	5582.6	0.00	197	5582.6	**5582.6**	87	5582.6	**5582.6**
$i(3,2,90,1)$	obj1	2731.00	0.00	57	2731.00	**2731.00**	60	2731.00	**2731.00**
$i(3,2,90,1)$	obj2	27.00	0.00	72	27.00	**27.00**	84	27.00	**27.00**
$i(3,2,90,1)$	obj3	40.00	0.00	164	40.00	**40.00**	68	40.00	**40.00**
$i(3,2,90,1)$	obj4	4182.22	0.00	148	4182.22	**4182.22**	47	4182.22	**4182.22**

$i(3,2,90,2)$	obj1	2545.63	0.00	252	2545.63	**2545.63**	70	2545.63	**2545.63**
$i(3,2,90,3)$	obj1	2262.84	0.00	117	2262.84	**2262.84**	73	2262.84	**2262.84**
$i(3,2,90,4)$	obj1	2876.44	0.00	356	2876.44	**2876.44**	65	2876.44	**2876.44**
$i(3,2,90,5)$	obj1	3731.41	0.00	183	3731.41	**3731.41**	68	3731.41	**3731.41**
$i(3,3,90,1)$	obj1	4096.5	0.00	758	4096.5	**4096.5**	168	4096.5	**4096.5**
$i(3,3,90,1)$	obj2	39.00	0.00	2248	39.00	**39.00**	303	39.00	**39.00**
$i(3,3,90,1)$	obj3	60.00	0.00	1952	60.00	**60.00**	218	60.00	**60.00**
$i(3,3,90,1)$	obj4	6273.33	0.00	3486	6273.33	**6273.33**	171	6273.33	**6273.33**
$i(3,3,90,2)$	obj1	3818.45	0.00	4437	3818.45	**3818.45**	243	3818.45	**3818.45**
$i(3,3,90,3)$	obj1	3394.27	0.00	1792	3394.27	**3394.27**	241	3394.27	**3394.27**
$i(3,3,90,4)$	obj1	4314.66	0.00	1217	4314.66	**4314.66**	233	4314.66	**4314.66**
$i(3,3,90,5)$	obj1	5597.12	0.00	3008	5597.12	**5597.12**	244	5597.12	**5597.12**
$i^*(3,2,90,1)$	obj1	2731.00	3.46	14400	2731.00	2829.00	203	2829.00	**2829.00**
$i^*(3,2,90,1)$	obj2	27.00	0.00	62	27.00	**27.00**	83	27.00	**27.00**
$i^*(3,2,90,1)$	obj3	40.00	0.00	59	40.00	**40.00**	66	40.00	**40.00**
$i^*(3,2,90,1)$	obj4	4182.22	0.00	144	4182.22	**4182.22**	57	4182.22	**4182.22**
$i^*(3,2,90,2)$	obj1	2545.63	3.73	14400	2545.63	2644.33	240	2644.33	**2644.33**
$i^*(3,2,90,3)$	obj1	2262.84	3.15	14400	2262.84	2336.34	228	2336.34	**2336.34**
$i^*(3,2,90,4)$	obj1	2876.44	0.34	14400	2876.44	2886.24	243	2886.24	**2886.24**
$i^*(3,2,90,5)$	obj1	3731.41	3.57	14400	3731.41	3869.73	271	3869.73	**3869.73**
$i(4,2,91,1)$	obj1	2740.08	3.45	14400	2740.08	2838.08	264	2838.08	**2838.08**
$i(4,2,91,1)$	obj2	32.00	0.00	752	32.00	**32.00**	122	32.00	**32.00**
$i(4,2,91,1)$	obj3	48.00	0.00	855	48.00	**48.00**	77	48.00	**48.00**
$i(4,2,91,1)$	obj4	4236.56	0.00	65	4236.56	**4236.56**	68	4236.56	**4236.56**
$i(4,2,91,2)$	obj1	2551.36	3.72	14400	2551.36	2650.06	329	2650.06	**2650.06**
$i(4,2,91,3)$	obj1	2268.23	3.14	14400	2268.23	2341.73	271	2341.73	**2341.73**
$i(4,2,91,4)$	obj1	2881.76	0.34	14400	2881.76	2891.56	279 1	2891.561	**2891.56**
$i(4,2,91,5)$	obj1	3741.09	3.57	14400	3741.09	3879.41	323	3879.41	**3879.41**
Average Computing times					3492			131	

Table 3: With and without new branching rule and heuristic

For the instances with two doors and two paths, the general solver is slightly better in all instances except one. But with more doors or paths, the new method obtains better solution times. Therefore, for larger instances our proposed method performs much better. Particularly, it can solve many instances that the general solver cannot solve within the time limit of 4 hours. These are the instances where a gap between linear relaxation at the root node and optimal solution still exists.

6 Conclusions

In this paper we have proposed for the first time in the literature an MILP formulation to design the DMS of an aircraft while optimizing a certain criterion (e.g., cost or weight of the system). It is a network system with multiple functions and k-redundancy for the functions.
Because the MILP formulation is not easily solvable with a standard solver like CPLEX, we introduced a new branching rule and we proposed a heuristic. Both are specialized for the model and through computational tests we were able to see that the DMS problem with redundancy can be solved much more efficiently. Problem instances which could not be solved in several hours can now be solved in minutes by using the new branching rule and heuristic.
Regarding future research, it must be noted that both functions in the DMS transport data through a hardware network system. In the current model, the speed of the data transportation was not considered. But the speed of the data through the system is also important in safety systems. Therefore, further research should include adding these requirements to the model. The corresponding constraints will be non-linear, which is a considerable increase in difficulty. Another aspect for future research is to consider reliability on the network system by giving to

each of the components a certain failure probability and then include in the model constraints that guarantee that the failure probability of the system does not exceed a certain threshold. Difficulties not only arise because non-linear constraints have to be added, but also because the number of constraints and variables will increase considerably.

Acknowledgements

The research of Sergio García has been funded by Fundación Séneca (project 19320/PI/14). Lukas Schäfer is funded by an EPSRC Industrial CASE studentship in partnership with Airbus Group.

References

[1] T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007.

[2] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[3] T. Eilam-Tzoreff. The disjoint shortest paths problem. *Discrete Applied Mathematics*, 85(2):113–138, 1998.

[4] T. Gomes, J. Craveirinha, and L. Jorge. An effective algorithm for obtaining the minimal cost pair of disjoint paths with dual arc costs. *Computers & Operations Research*, 36(5):1670–1682, 2009.

[5] M. Grötschel, C. Monma, and M. Stoer. Design of survivable networks. In *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 617–672. Elsevier, 1995.

[6] L. Guo, H. Shen, and K. Liao. *Improved Approximation Algorithms for Computing k Disjoint Paths Subject to Two Constraints*, pages 325–336. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[7] F. Iqbal and F. A. Kuipers. Disjoint paths in networks. *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 77–86, 2015.

[8] J. Kalvenes, J. Kennington, and E. Olinick. Base station location and service assignments in W–CDMA networks. *INFORMS Journal on Computing*, 18(3):366–376, 2006.

[9] F. Kuipers. An overview of algorithms for network survivability. *ISRN Communications and Networking*, 2012:1–19, 2012.

[10] Z. Liang, W. A. Chaovalitwongse, M. Cha, and S. B. Moon. Redundant multicast routing in multilayer networks with shared risk resource groups: Complexity, models and algorithms. *Computers & Operations Research*, 37(10):1731–1739, 2010.

[11] I. Rodríguez-Martín, J. Salazar-González, and H. Yaman. A branch-and-cut algorithm for two-level survivable network design problems. *Computers & Operations Research*, 67:102–112, 2016.

[12] F. S. Salman, R. Ravi, and J. N. Hooker. Solving the capacitated local access network design problem. *INFORMS Journal on Computing*, 20(2):243–254, 2008.

[13] J. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

[14] L. A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998.

A Negative validity example

In this example, we have $|D| = 3$, $|P| = 2$ and $|F| = 2$. There are four different unit types $U = \{U_1, U_2, U_3, U_4\}$. A unit of type U_1 cannot be used for different doors. All needed information of the units is given in Table 4 and 5 for functions f_1 and f_2, respectively.

Unit types	Model	E_{um}	E_{fu}^{in}	E_{fu}^{out}	$T_u^{f,in}$	$T_u^{f,out}$	$C_f^+(u)$	$C_f^-(u)$
U_1	M_1	2	1	1	1	1	$\{U_2\}$	$\emptyset$
U_2	M_1	4	1	1	1	1	$\{U_3\}$	$\emptyset$
U_2	M_2	5	1	1	1	1	$\{U_3\}$	$\emptyset$
U_3	M_1	8	1	1	1	1	$\emptyset$	$\emptyset$

Table 4: Unit parameters and corresponding sets for function f_1.

Unit types	Model	E_{um}	E_{fu}^{in}	E_{fu}^{out}	$T_u^{f,in}$	$T_u^{f,out}$	$C_f^+(u)$	$C_f^-(u)$
U_2	M_1	4	1	1	1	1	$\emptyset$	$\{U_3\}$
U_2	M_2	5	1	1	1	1	$\emptyset$	$\{U_3\}$
U_3	M_1	8	1	1	1	1	$\emptyset$	$\emptyset$
U_4	M_1	4	1	1	1	1	$\emptyset$	$\{U_2\}$

Table 5: Unit parameters and corresponding sets for function f_2.

Tables 8 to 12 give a feasible linear relaxation solution for the small example and Figure 5 illustrates it in a graph. To distinguish the different functions in Figure 5 we have used thicker arrows for function f_2. As can be seen, 2-redundancy is not given and constraint (12) is fully satisfied for units of type $U2$. Even though the system is feasible in the linear relaxation, for this combination of units there are no cable connections which are feasible in the integer formulation and unit type U_2. If such a case as here is discovered through the validity test, we can then branch on the corresponding t_{um} variables. In this example, we would branch with $t_{U_2,M_1} \geq 3$ and $t_{U_2,M_1} \leq 1$ or $t_{U_2,M_2} \geq 2$ and $t_{U_2,M_2} \leq 0$.

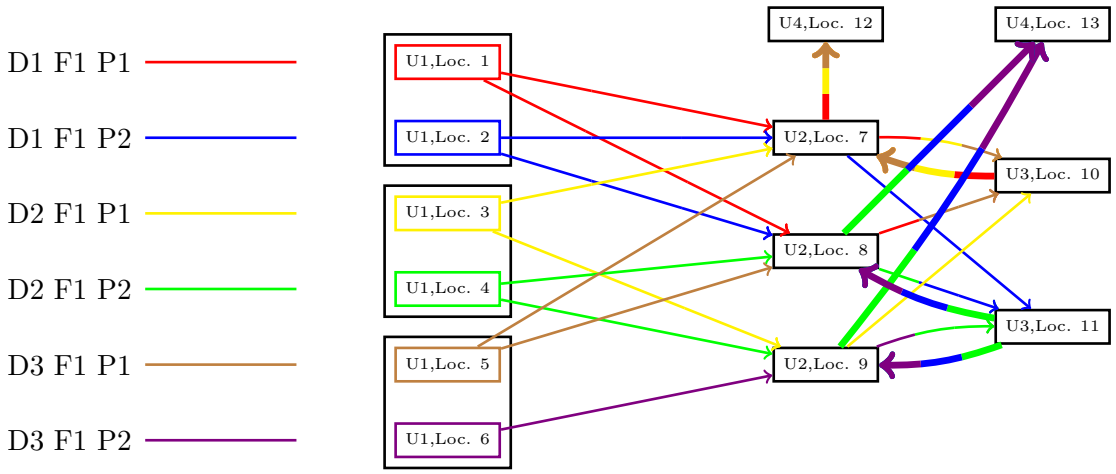

Figure 5: Illustration of liner relaxation solution. Solution can be found in Tables 9 and 10.

i	1	2	3	4	5	6	7	8	9	10	11	12	13
u	U_1	U_1	U_1	U_1	U_1	U_1	U_2	U_2	U_2	U_3	U_3	U_4	U_4
t_{i,u,M_1}	1	1	1	1	1	1	0	1	1	1	1	1	1
t_{i,u,M_2}	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 6: Linear relaxation solution values for location variables of the overall system.

i	1	1	2	2	3	3	4	4	5	5	6	7	7	8	8	9	9
j	7	8	7	8	7	9	8	9	7	8	9	10	11	10	11	10	11
u	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_2	U_2	U_2	U_2	U_2	U_2
$\hat{u}$	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_3	U_3	U_3	U_3	U_3	U_3
$x_{iju\hat{u}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1	0.5	0.5	0.5	0.5	0.5	1

Table 7: Linear relaxation solution values for connection variables of the overall system.

i	7	7	8	8	9	9	10	10	10	11	11	11
j	12	13	12	13	12	13	7	8	9	7	8	9
u	U_2	U_2	U_2	U_2	U_2	U_2	U_3	U_3	U_3	U_3	U_3	U_3
$\hat{u}$	U_4	U_4	U_4	U_4	U_4	U_4	U_2	U_2	U_2	U_2	U_2	U_2
$x_{iju\hat{u}}$	1	0	0	0.5	0	0.5	1	0	0	0	0.5	0.5

Table 8: Linear relaxation solution values for connection variables of the overall system.

i	u	t_{d_1,f_1,p_1}	t_{d_1,f_1,p_2}	t_{d_2,f_1,p_1}	t_{d_2,f_1,p_2}	t_{d_3,f_1,p_1}	t_{d_3,f_1,p_2}
1	U_1	1	0	0	0	0	0
2	U_1	0	1	0	0	0	0
3	U_1	0	0	1	0	0	0
4	U_1	0	0	0	1	0	0
5	U_1	0	0	0	0	1	0
6	U_1	0	0	0	0	0	1
7	U_2	0.5	0.5	0.5	0	0.5	0
8	U_2	0.5	0.5	0	0.5	0.5	0
9	U_2	0	0	0.5	0.5	0	1
10	U_3	1	0	1	0	1	0
11	U_3	0	1	0	1	0	1

Table 9: Linear relaxation solution values for location variables and function f_1.

i	j	u	$\hat{u}$	$x_{iju\hat{u}}$	x_{d_1,f_1,p_1}	x_{d_1,f_1,p_2}	x_{d_2,f_1,p_1}	x_{d_2,f_1,p_2}	x_{d_3,f_1,p_1}	x_{d_3,f_1,p_2}
1	7	U_1	U_2	0.5	0.5	0	0	0	0	0
1	8	U_1	U_2	0.5	0.5	0	0	0	0	0
2	7	U_1	U_2	0.5	0	0	0	0	0	0
2	8	U_1	U_2	0.5	0	0.5	0	0	0	0
3	7	U_1	U_2	0.5	0	0	0.5	0	0	0
3	9	U_1	U_2	0.5	0	0	0.5	0	0	0
4	8	U_1	U_2	0.5	0	0	0	0.5	0	0
4	9	U_1	U_2	0.5	0	0	0	0.5	0	0
5	7	U_1	U_2	0.5	0	0	0	0	0.5	0
5	8	U_1	U_2	0.5	0	0	0	0	0.5	0
6	9	U_1	U_2	1	0	0	0	0	0	1
7	10	U_2	U_3	0.5	0.5	0	0.5	0	0.5	0
7	11	U_2	U_3	0.5	0	0.5	0	0	0	0
8	10	U_2	U_3	0.5	0.5	0	0	0	0.5	0
8	11	U_2	U_3	0.5	0	0.5	0	0.5	0	0
9	10	U_2	U_3	0.5	0	0	0.5	0	0	0
9	11	U_2	U_3	1	0	0	0	0.5	0	1

Table 10: Linear relaxation solution values for connection variables.

i	u	t_{d_1,f_1,p_1}	t_{d_1,f_1,p_2}	t_{d_2,f_1,p_1}	t_{d_2,f_1,p_2}	t_{d_3,f_1,p_1}	t_{d_3,f_1,p_2}
7	U_2	0.5	0.5	0.5	0	0.5	0
8	U_2	0.5	0.5	0	0.5	0.5	0
9	U_2	0	0	0.5	0.5	0	1
10	U_3	1	0	1	0	1	0
11	U_3	0	1	0	1	0	1
12	U_4	1	0	1	0	1	0
13	U_4	0	1	0	1	0	1

Table 11: Linear relaxation solution values for location variables and function f_2.

i	j	u	$\hat{u}$	x_{d_1,f_1,p_1}	x_{d_1,f_1,p_2}	x_{d_2,f_1,p_1}	x_{d_2,f_1,p_2}	x_{d_3,f_1,p_1}	x_{d_3,f_1,p_2}
7	12	U_2	U_4	1	0	1	0	1	0
7	13	U_2	U_4	0	0	0	0	0	0
8	12	U_2	U_4	0	0	0	0	0	0
8	13	U_2	U_4	0	0.5	0	0.5	0	0.5
9	12	U_2	U_4	0	0	0	0	0	0
9	13	U_2	U_4	0	0.5	0	0.5	0	0.5
10	7	U_3	U_2	1	0	1	0	1	0
10	8	U_3	U_2	0	0	0	0	0	0
10	9	U_3	U_2	0	0	0	0	0	0
11	7	U_3	U_2	0	0	0	0	0	0
11	8	U_3	U_2	0	0.5	0	0.5	0	0.5
11	9	U_3	U_2	0	0.5	0	0.5	0	0.5

Table 12: Linear relaxation solution values for connection variables for function f_2.